DATABASE MANAGEMENT AND DATABASE ANALYTICS

XYZ Healthcare Organization NoSQL Database

Abinaya Sundari Panneerselvam

Scenario

The scenario focus on how XYZ healthcare organization that manages several hospitals, clinics, and diagnostic facilities throughout an area. Millions of individuals' medical records, including their personal data, medical background, test results, treatments, billing, and prescriptions, are maintained by this institution. This organization must maintain large set of details about each patient. So, often they are facing issues with managing and accessing this data in an efficient way. Here, the data is growing rapidly, the volume and complexity of the data also increasing. The traditional relational database struggling to handle this sizeable data. In this case, the healthcare organization is searching for a more adaptable and versatile solution to store and manage their patient data. They seek a database system that supports real-time data processing and analytics and can manage large volumes of data and queries. The organization also needs to be able to track and analyze trends and patterns in the data to inform decision-making and improve patient care.

NoSQL DB Form:

The NoSQL database support speedy and flexible data manipulation and can handle enormous quantity of unstructured or semi-structured data unlike relational database, can handle only structured data. It has various features such as indexing, aggregation pipelines and horizontal scaling to support real-time data processing and analytics. So, the NoSQL database would be the correct scenario for this healthcare organization to meet its requirements.

MongoDB is such a NoSQL database that might be employed in this situation. MongoDB is a well-known document-oriented database that stores data in documents that resemble JSON, enabling a flexible and scalable data model. Through its aggregation pipeline and indexing features, it also offers real-time data processing and analytics.

Why a NoSQL Solution Would Work Best:

In this scenario, the NoSQL solution like MongoDB would work best for the healthcare organization for several reasons. First, the firm has a wide range of sophisticated data requirements. The patient data is comprised of a variety of variables that can be challenging to model in a conventional relational database, including personal information, medical history, test results, therapies, and prescriptions. On the other hand, a NoSQL database like MongoDB enables a more flexible data model, allowing each patient record to be represented as a single document with a range of fields. Instead of having to divide the complex data across various tables and relationships, it can now be stored and managed in a single collection.

Second, the organization's data volume is also substantial and expanding. Millions of patient records are handled by the healthcare institution, and more data is anticipated to be generated over time. Such a large amount of data may be difficult for a standard relational database to

manage, especially if real-time data processing and analytics are required. On the other hand, a NoSQL database like MongoDB is built to manage enormous volumes of data and queries and can extend horizontally to support increasing data volume and workloads.

Third, the organization's data requirements are dynamic and real-time. To support clinical decision-making and population health management, the healthcare organization must access and update patient data in real-time and run analytics on it. A standard relational database might not be able to meet the organization's demands for real-time data processing and analytics. On the other hand, a NoSQL database like MongoDB offers quick and adaptable data manipulation through its aggregation pipeline and indexing features, making it a good choice for addressing real-time data requirements.

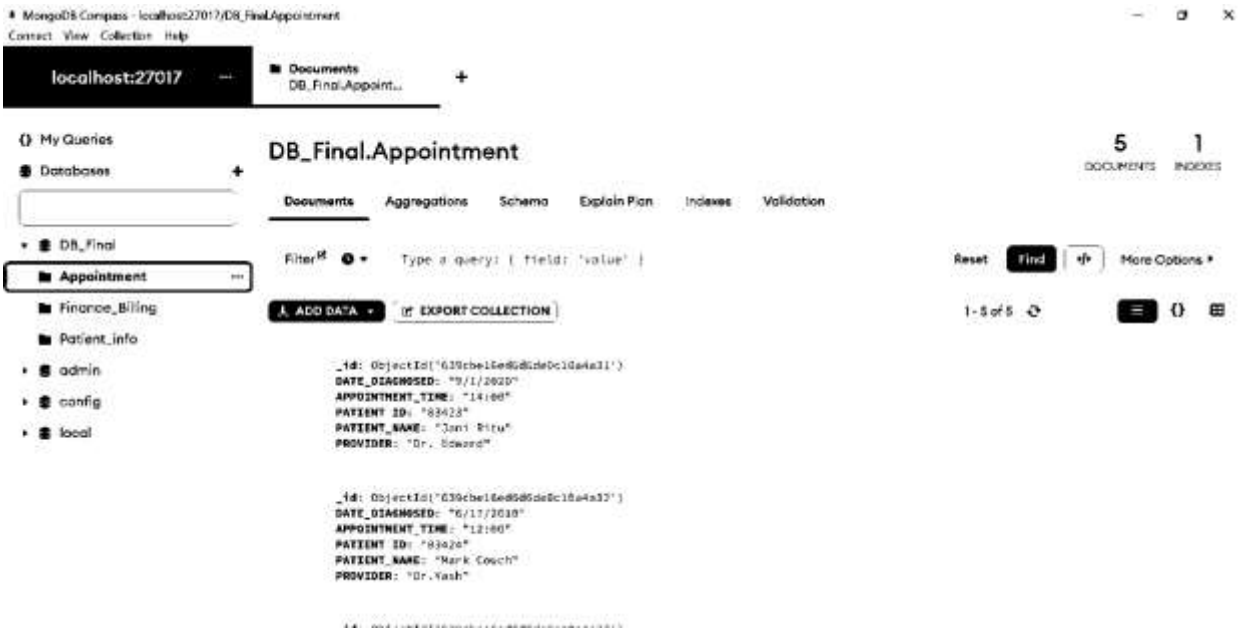
Barebones Prototype:

Here, the barebones prototype of database in MongoDB Compass:

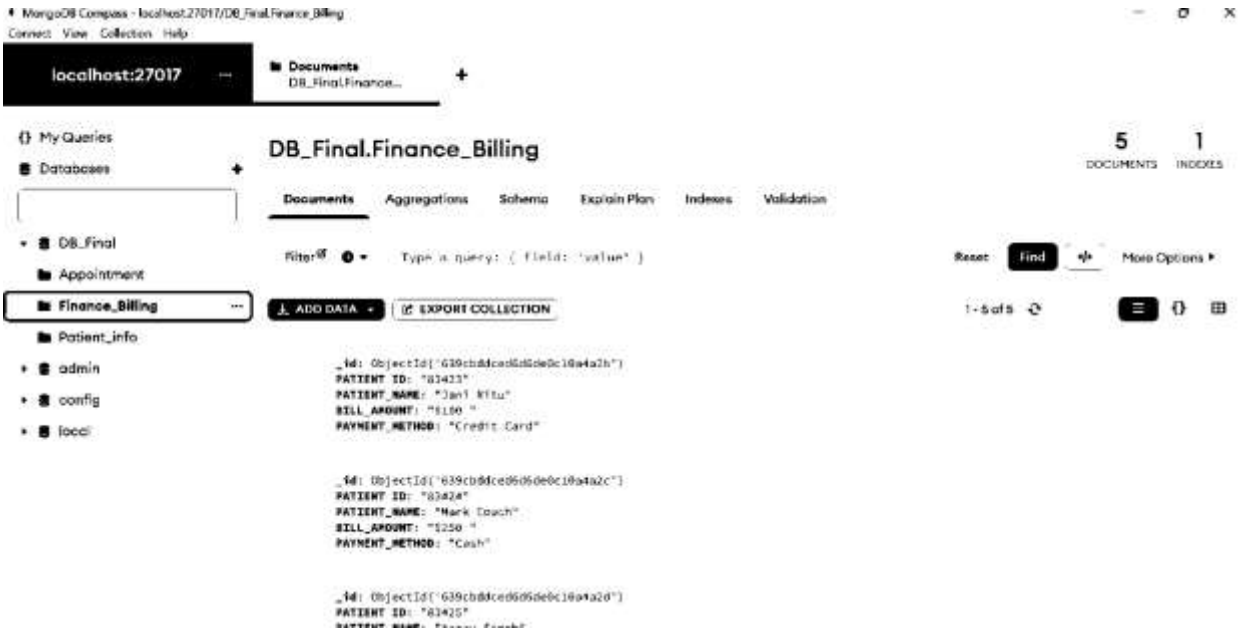
The XYZ healthcare database has three collection Patient_info, Appointment and Finance_Billing.

The collection, Patient_info has personal details like patient’s id, patient’s name, age, address, and medical conditions like disease and allergy of the patient.

Appointment has information about appointment time, appointment date and provider.



Finance_Billing has details about bill amount and payment method.



The document view of the data can be seen above. Here, XYZ healthcare organization maintaining three different records like patients' details, appointment details and billing in the same database. The usage of MongoDB helps the XYZ organization deals with large volume of data as a single database.

We can also perform lot of operations like filter, project, and sort in MongoDB.

Filter:

In the above screenshot, the patients with blood pressure are extracted separately using filter operation. The "Filter" feature allows us to specify criteria for selecting documents from a collection. This can be useful if we want to focus on a specific subset of documents.

The concept of aggregation is one of the most useful functions in MongoDB. One can perform lookup, addfield, group, count, match, etc., using aggregation function.

Lookup:

The "lookup" stage of aggregate pipeline allows us to perform a left outer join on two collections and merge them into a single collection. It does this by creating a new array field for the joined documents and populating it with the documents from the "join" collection that match the join condition.

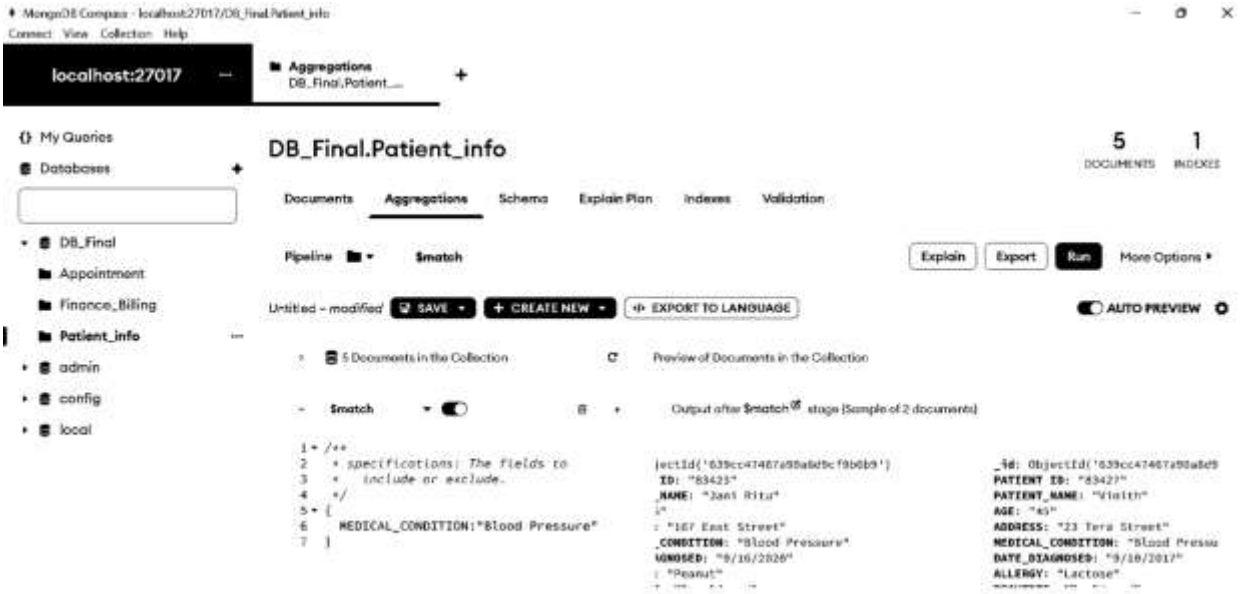
When we need to combine two collections together but don't want to de-normalize the data by combining them into one, the lookup function comes in handy. When we need to access data from both collections in a single query but still want to preserve the integrity and independence of the two collections, this can be quite helpful.

In this scenario, the XYZ healthcare organization with a wide range of information requirements, the lookup function could be used to join the "Appointment" collection with the "Patient_info" collection to retrieve information about the appointment time of the patient.

Add field:

The Add Field stage of aggregate pipeline allows us to add a new field to a document in a collection. Here, XYZ healthcare organization add a new field to their “Finance_Billing” collection called Branch_ID, which is the branch number of hospital.

Match:



The "Match" stage of the aggregate pipeline allows us to specify criteria for selecting documents from a collection. The match stage operates like a filter, allowing us to focus on a specific subset of documents based on certain criteria. In this scenario, the XYZ healthcare organization use match function to match patient with same medical condition as, patient with blood pressure.

Sorting:

The "sort" stage of the aggregate pipeline allows us to sort the documents in a collection based on one or more fields. This can be useful if we want to organize the documents in a specific order.

In the below screenshot, the age of the patients is sorted in ascending order by using sort function. This function helps us to find what is the minimum age of patients when we are doing analysis. We can even sort in descending order.